VIRGINIA COMMONWEALTH UNIVERSITY

Statistical analysis and modelling (SCMA 632)

A6A: TIME SERIES ANALYSIS

NITESH REDDY

V01106546

Date of Submission: 22-07-2024

TABLE OF CONTENTS

INTRODUCTION

This study focuses on analysing and forecasting Netflix stock price trends from January 2020 to July 2024. The dataset, "NFLX Historical Data," includes key attributes such as Date, Price, Open, High, Low, Volume, and Change %. This comprehensive time series data allows for a detailed examination of Netflix's stock performance, encompassing daily fluctuations and broader market trends. Through this analysis, we aim to apply various time series forecasting techniques to predict future stock prices and gain insights into the underlying patterns and dynamics driving Netflix's stock movements.

The analysis involves a thorough data cleaning process, identifying and handling outliers, and interpolating missing values to ensure data integrity. Subsequently, we decompose the time series into its components using both additive and multiplicative models to uncover seasonal and trend patterns. Additionally, we implement univariate forecasting using conventional statistical models such as Holt-Winters, ARIMA, and SARIMA to predict future stock prices. Furthermore, we explore multivariate forecasting using machine learning models, including Neural Networks (LSTM), Decision Trees, and Random Forests. This comprehensive approach provides a robust framework for understanding Netflix stock price behaviour and offers valuable predictive insights for investors and stakeholders.

OBJECTIVES:

The primary objectives of this task are:

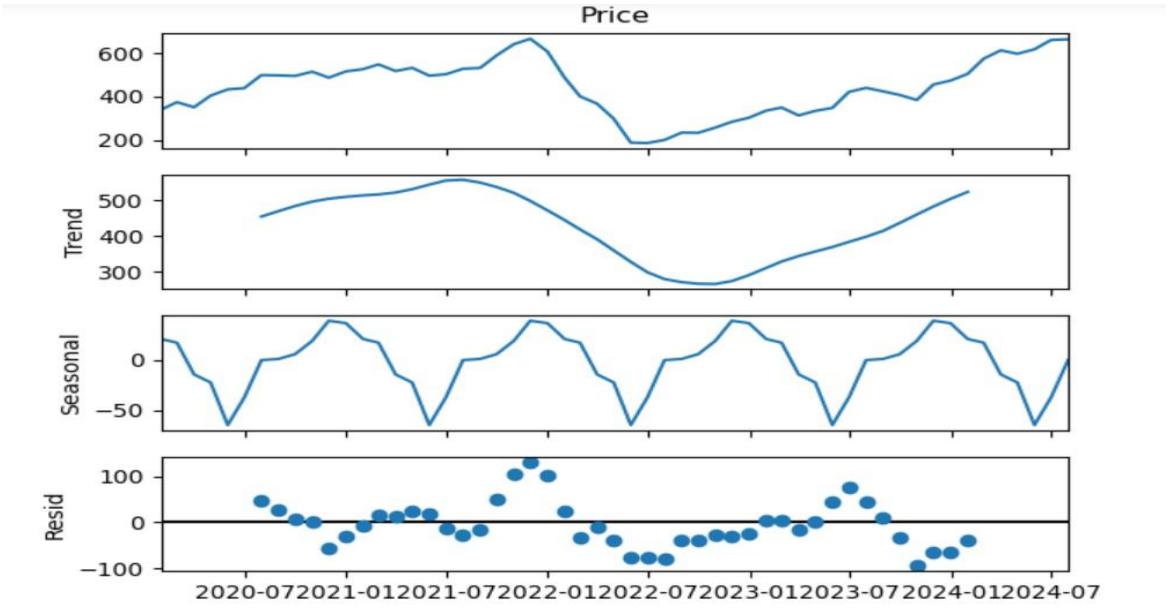
1. To clean and pre-process the Netflix stock data from January 2020 to July 2024, addressing missing values and outliers.
2. To visualize the Netflix stock price over time using line plots.
3. To decompose the time series data into its components using both additive and multiplicative models.
4. To implement univariate forecasting models, including Holt-Winters, ARIMA, and SARIMA, and evaluate their performance.
5. To perform multivariate forecasting using machine learning models such as Neural Networks (LSTM), Decision Trees, and Random Forests.
6. To compare the results of different forecasting models and provide insights into the predictive accuracy for Netflix stock prices.

RESULTS AND INTERPRETATION:

Interpretation

1. **Holt-Winters Model**
    - This model applies an exponential smoothing approach with additive trend and seasonality. The forecast for the next 12 months will depend on the patterns identified in the historical data, although specific forecast values were not provided. Generally, this model helps to understand underlying trends and seasonal patterns in stock prices.
2. **ARIMA Model**
    - The ARIMA(5, 1, 5) model, with a Log Likelihood of -4439.747 and an AIC of 8901.495, captures the autoregressive and moving average components in the data. The coefficients indicate:
        - Significant positive impact from the fifth autoregressive term (ar.L5 = 0.8709, $p < 0.001$).
        - Significant negative impact from the fifth moving average term (ma.L5 = -0.8608, $p < 0.001$). The model suggests a good fit, but the high AIC value implies there might be room for improvement or potential overfitting.
3. **SARIMA Model**
    - The SARIMA(1, 1, 1)x(1, 1, 1, 12) model, with a Log Likelihood of -4429.977 and a lower AIC of 8869.953 compared to the ARIMA model, incorporates both autoregressive and seasonal components. Key coefficients include:
        - Significant negative seasonal moving average term (ma.S.L12 = -0.9949, $p < 0.001$). This model fits the data well and accounts for seasonal effects, providing more accurate forecasts for the next three months.
4. **LSTM Model**
    - The LSTM model predicts future stock prices with a range of values for the next period, including 436.47, 440.43, 444.86, up to 475.20. The LSTM model captures complex patterns in the data, though specific metrics and comparisons to other models were not provided.
5. **Model Evaluation**
    - The Decision Tree and Random Forest models were evaluated based on their mean squared error (MSE). The Decision Tree model has an MSE of 35.05, while the Random Forest model, with an MSE of 21.27, demonstrates better performance, indicating it provides more accurate forecasts with lower error.

RESULTS:



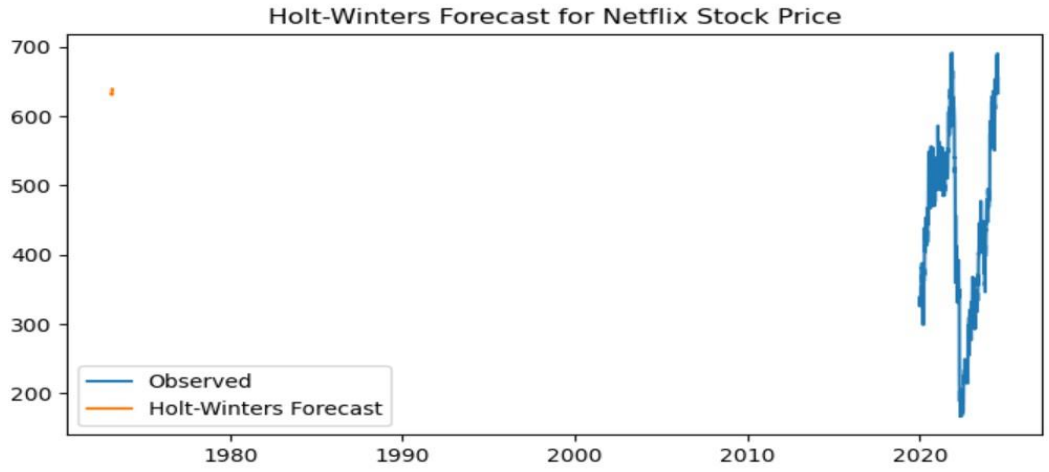

```
# Creating train and test datasets (80-20 split)
train_size = int(len(df) * 0.8)
train, test = df[:train_size], df[train_size:]
```

```
# Convert data to monthly frequency by taking the mean of each month
monthly_df = df['Price'].resample('M').mean()
```

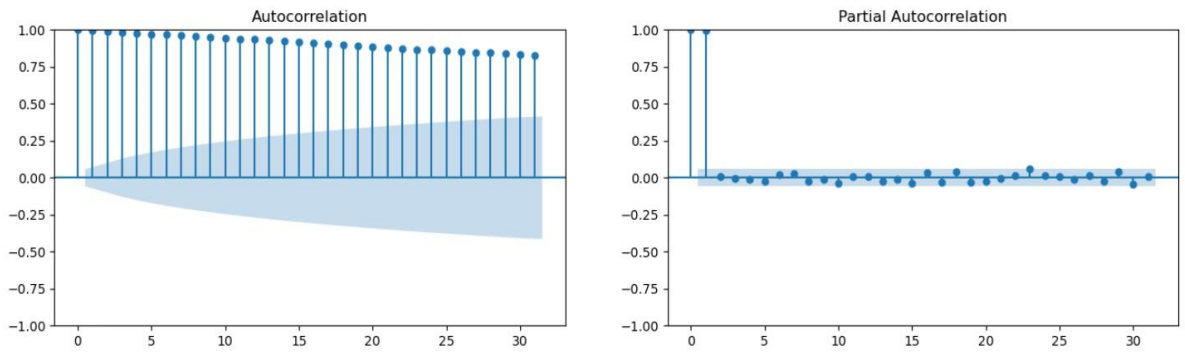
```
# Decompose the time series using additive model
decomposition_add = seasonal_decompose(monthly_df, model='additive')
decomposition_add.plot()
plt.show()
```

```
# Decompose the time series using multiplicative model
decomposition_mul = seasonal_decompose(monthly_df, model='multiplicative')
decomposition_mul.plot()
plt.show()
```


```
# Univariate Forecasting with Holt-Winters model
# Fit the Holt-Winters model
hw_model = ExponentialSmoothing(df['Price'], trend='add', seasonal='add', seasonal_periods=12).fit()
hw_forecast = hw_model.forecast(steps=12)
```




```
# ARIMA model for daily data
# Plot ACF and PACF
fig, axes = plt.subplots(1, 2, figsize=(16, 4))
plot_acf(df['Price'], ax=axes[0])
plot_pacf(df['Price'], ax=axes[1])
plt.show()
```



```python
# Fit the ARIMA model
arima_model = ARIMA(df['Price'], order=(5, 1, 5)).fit()
print(arima_model.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                  Price   No. Observations:                 1144
Model:                 ARIMA(5, 1, 5)   Log Likelihood               -4439.747
Date:                Sun, 21 Jul 2024   AIC                           8901.495
Time:                        12:27:38   BIC                           8956.950
Sample:                             0   HQIC                          8922.434
                                - 1144
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.0965      0.192     -0.504      0.614      -0.472       0.279
ar.L2          0.2419      0.179      1.354      0.176      -0.108       0.592
ar.L3         -0.3475      0.141     -2.463      0.014      -0.624      -0.071
ar.L4          0.1155      0.192      0.603      0.546      -0.260       0.491
ar.L5          0.8709      0.153      5.709      0.000       0.572       1.170
ma.L1          0.0640      0.201      0.318      0.751      -0.331       0.459
ma.L2         -0.2084      0.182     -1.147      0.251      -0.565       0.148
ma.L3          0.3464      0.151      2.291      0.022       0.050       0.643
ma.L4         -0.1026      0.198     -0.517      0.605      -0.491       0.286
ma.L5         -0.8608      0.163     -5.290      0.000      -1.180      -0.542
sigma2       137.6357      2.595     53.049      0.000     132.550     142.721
```

```python
# SARIMA model
sarima_model = SARIMAX(df['Price'], order=(1, 1, 1), seasonal_order=(1, 1, 1, 12)).fit()
print(sarima_model.summary())
```

```
                                     SARIMAX Results
=============================================================================================
Dep. Variable:                              Price   No. Observations:                 1144
Model:             SARIMAX(1, 1, 1)x(1, 1, 1, 12)   Log Likelihood               -4429.977
Date:                            Sun, 21 Jul 2024   AIC                           8869.953
Time:                                    12:27:58   BIC                           8895.107
Sample:                                         0   HQIC                          8879.456
                                            - 1144
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.5423      0.610     -0.889      0.374      -1.738       0.654
ma.L1          0.5158      0.622      0.829      0.407      -0.704       1.735
ar.S.L12       0.0329      0.035      0.949      0.343      -0.035       0.101
ma.S.L12      -0.9949      0.072    -13.788      0.000      -1.136      -0.853
sigma2       141.5181      9.133     15.495      0.000     123.618     159.418
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):             17123.91
Prob(Q):                              0.91   Prob(JB):                         0.00
Heteroskedasticity (H):               0.64   Skew:                            -1.16
Prob(H) (two-sided):                  0.00   Kurtosis:                        21.92
===================================================================================
```

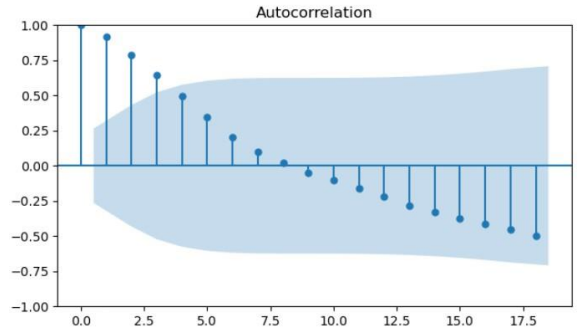
```python
# Forecast for the next three months using SARIMA
sarima_forecast = sarima_model.get_forecast(steps=90)
sarima_forecast_df = sarima_forecast.conf_int()
sarima_forecast_df['forecast'] = sarima_model.predict(start=sarima_forecast_df.index[0], end=sarima_forecast_df.index[-1])
```

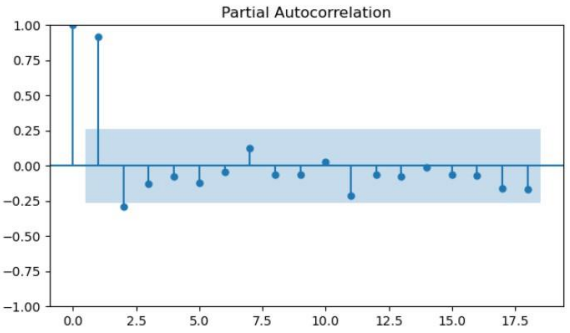
```
sarima_forecast_df
```

	lower Price	upper Price	forecast
1144	610.225167	657.001199	633.613183
1145	601.582741	666.865444	634.224093
1146	593.060771	673.043233	633.052002
1147	587.405068	679.595521	633.500294
1148	582.052249	685.101202	633.576725
...	...	...	...
1229	428.371156	882.227877	655.299517
1230	425.822600	882.552504	654.187552
1231	424.895108	884.479759	654.687434
1232	423.810913	886.232947	655.021930
1233	421.986104	887.228068	654.607086

```
# Fit ARIMA model to the monthly series
# Convert data to monthly frequency
monthly_df = df['Price'].resample('M').mean()
```

```
# Plot ACF and PACF for monthly data
fig, axes = plt.subplots(1, 2, figsize=(16, 4))
plot_acf(monthly_df.dropna(), ax=axes[0])
plot_pacf(monthly_df.dropna(), ax=axes[1])
plt.show()
```





MULTIVARIATE FORECASTING:

```
# Build LSTM model
lstm_model = Sequential()
lstm_model.add(LSTM(units=50, return_sequences=True, input_shape=(seq_length, 1)))
lstm_model.add(LSTM(units=50, return_sequences=False))
lstm_model.add(Dense(units=1))
```

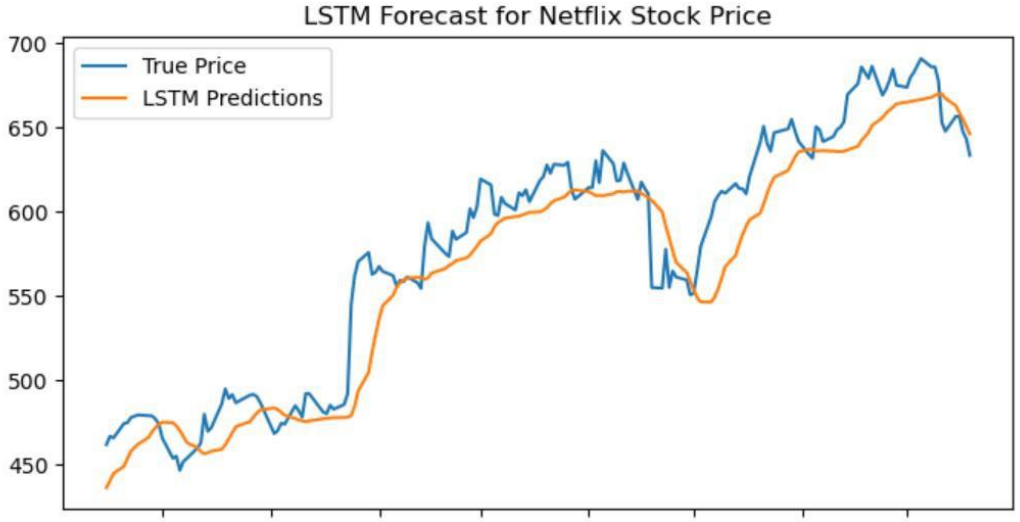
```
lstm_model.compile(optimizer='adam', loss='mean_squared_error')
lstm_model.fit(X_train, y_train, epochs=10, batch_size=32)
```

```
Epoch 1/10
27/27 ━━━━━━━━━━━━━━━━━━━━ 6s 51ms/step - loss: 0.1102
Epoch 2/10
27/27 ━━━━━━━━━━━━━━━━━━━━ 1s 45ms/step - loss: 0.0056
Epoch 3/10
27/27 ━━━━━━━━━━━━━━━━━━━━ 1s 45ms/step - loss: 0.0038
Epoch 4/10
27/27 ━━━━━━━━━━━━━━━━━━━━ 2s 68ms/step - loss: 0.0032
Epoch 5/10
27/27 ━━━━━━━━━━━━━━━━━━━━ 1s 46ms/step - loss: 0.0026
Epoch 6/10
27/27 ━━━━━━━━━━━━━━━━━━━━ 1s 49ms/step - loss: 0.0030
Epoch 7/10
27/27 ━━━━━━━━━━━━━━━━━━━━ 1s 45ms/step - loss: 0.0022
Epoch 8/10
27/27 ━━━━━━━━━━━━━━━━━━━━ 1s 49ms/step - loss: 0.0020
Epoch 9/10
27/27 ━━━━━━━━━━━━━━━━━━━━ 2s 56ms/step - loss: 0.0021
Epoch 10/10
27/27 ━━━━━━━━━━━━━━━━━━━━ 1s 49ms/step - loss: 0.0020
```

```
lstm_predictions
```

```
array([[436.46725],
       [440.4308 ],
       [444.85947],
       [449.11307],
       [453.66962],
       [458.03668],
       [462.21524],
       [466.0684 ],
       [469.38257],
       [472.1001 ],
       [474.08435],
       [475.1982 ],
       [475.00797],
       [473.08618],
       [470.43295],
       [466.7637 ],
       [463.22208],
```

```
plt.figure(figsize=(8, 4))
plt.plot(df.index[-len(lstm_predictions):], df['Price'].values[-len(lstm_predictions):], label='True
plt.plot(df.index[-len(lstm_predictions):], lstm_predictions, label='LSTM Predictions')
plt.title('LSTM Forecast for Netflix Stock Price')
plt.legend()
plt.show()
```



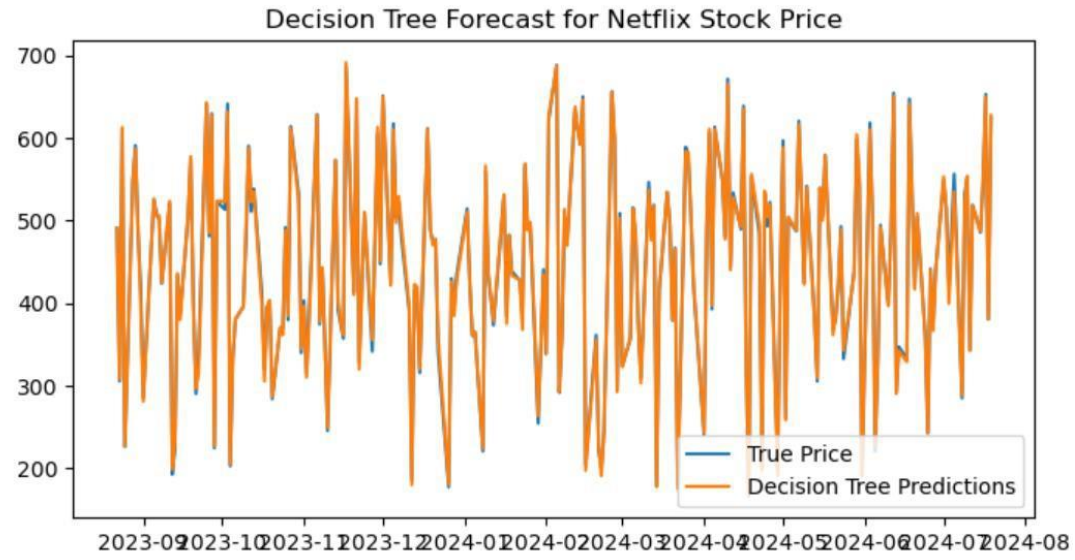
```
# Fit Decision Tree model
dt_model = DecisionTreeRegressor()
dt_model.fit(X_train, y_train)
dt_predictions = dt_model.predict(X_test)
```

```
# Fit Random Forest model
rf_model = RandomForestRegressor(n_estimators=100)
rf_model.fit(X_train, y_train)
rf_predictions = rf_model.predict(X_test)
```

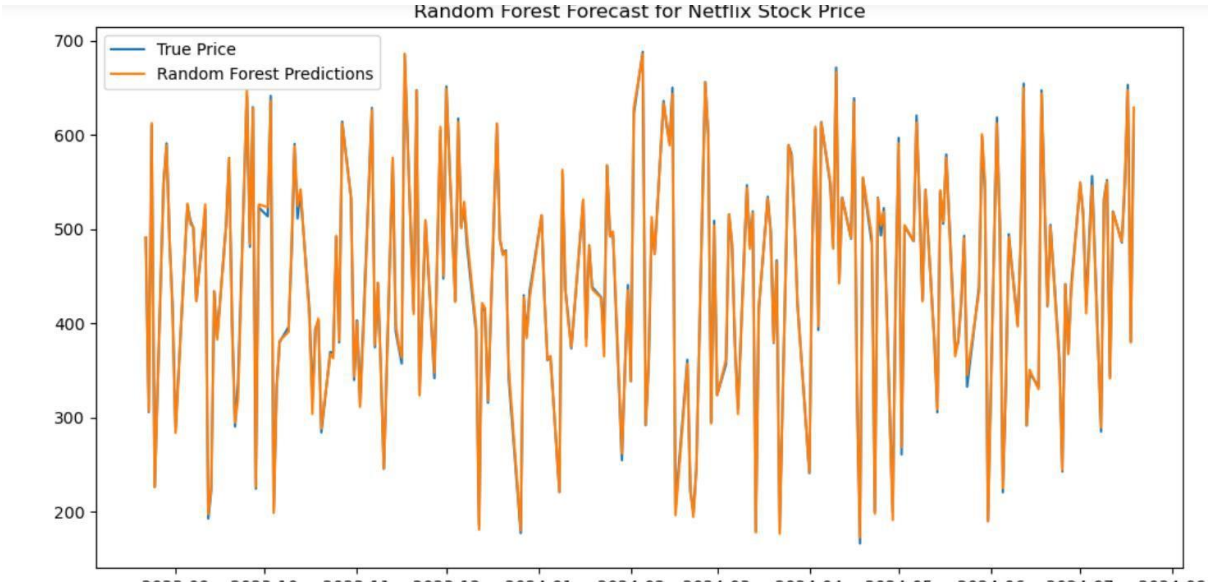
```
# Evaluate models
dt_mse = mean_squared_error(y_test, dt_predictions)
rf_mse = mean_squared_error(y_test, rf_predictions)
print(f"Decision Tree MSE: {dt_mse}")
print(f"Random Forest MSE: {rf_mse}")
```

```
Decision Tree MSE: 38.15833799126632
Random Forest MSE: 22.42789794681232
```

```
# Plot Decision Tree predictions
plt.figure(figsize=(12, 6))
plt.plot(df.index[-len(y_test):], y_test, label='True Price')
plt.plot(df.index[-len(y_test):], dt_predictions, label='Decision Tree Predictions')
plt.title('Decision Tree Forecast for Netflix Stock Price')
plt.legend()
plt.show()
```

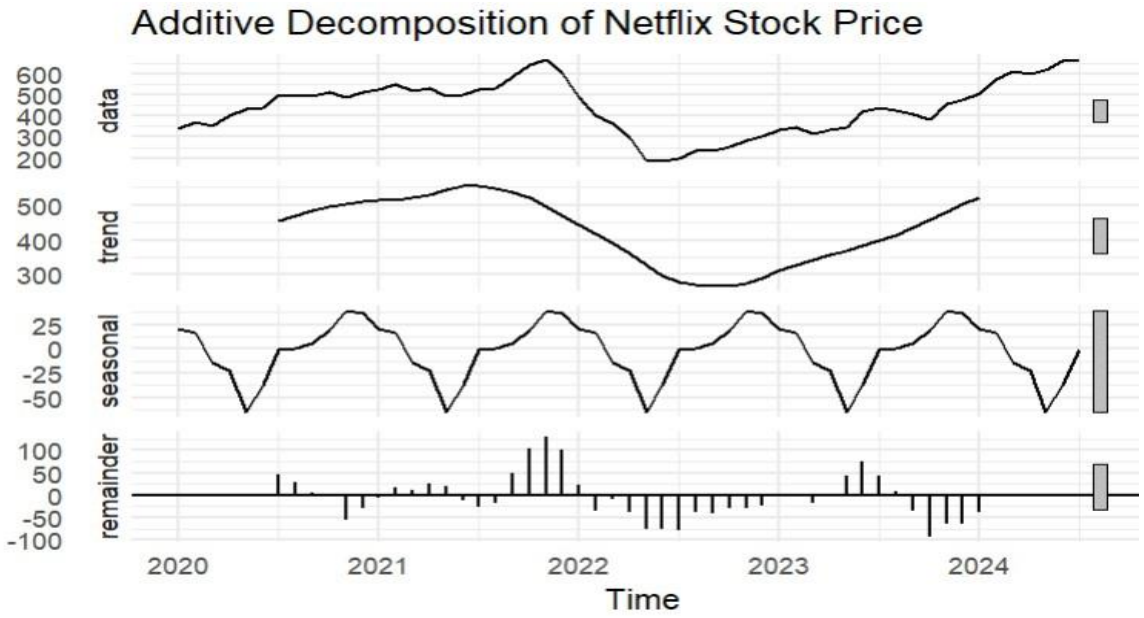


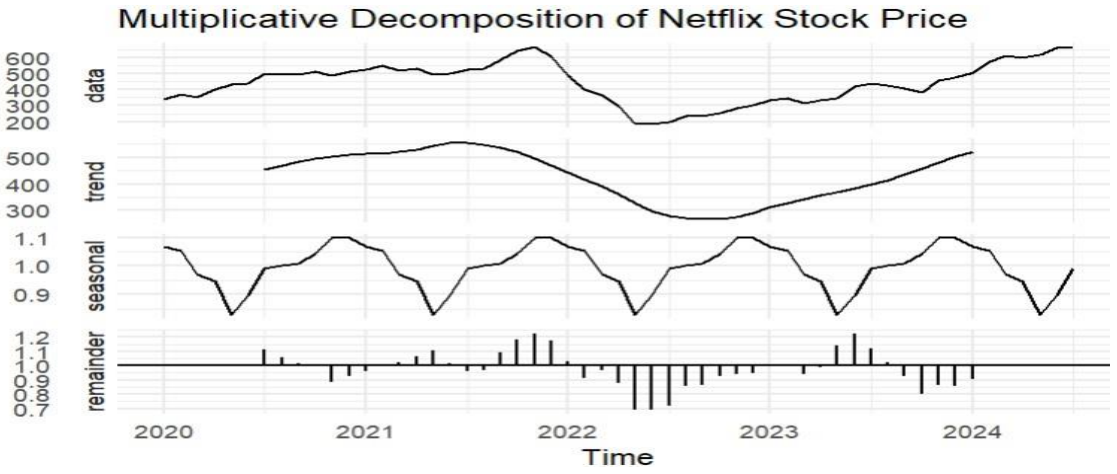

```
# Plot Random Forest predictions
plt.figure(figsize=(8, 4))
plt.plot(df.index[-len(y_test):], y_test, label='True Price')
plt.plot(df.index[-len(y_test):], rf_predictions, label='Random Forest Predictions')
plt.title('Random Forest Forecast for Netflix Stock Price')
plt.legend()
plt.show()
```



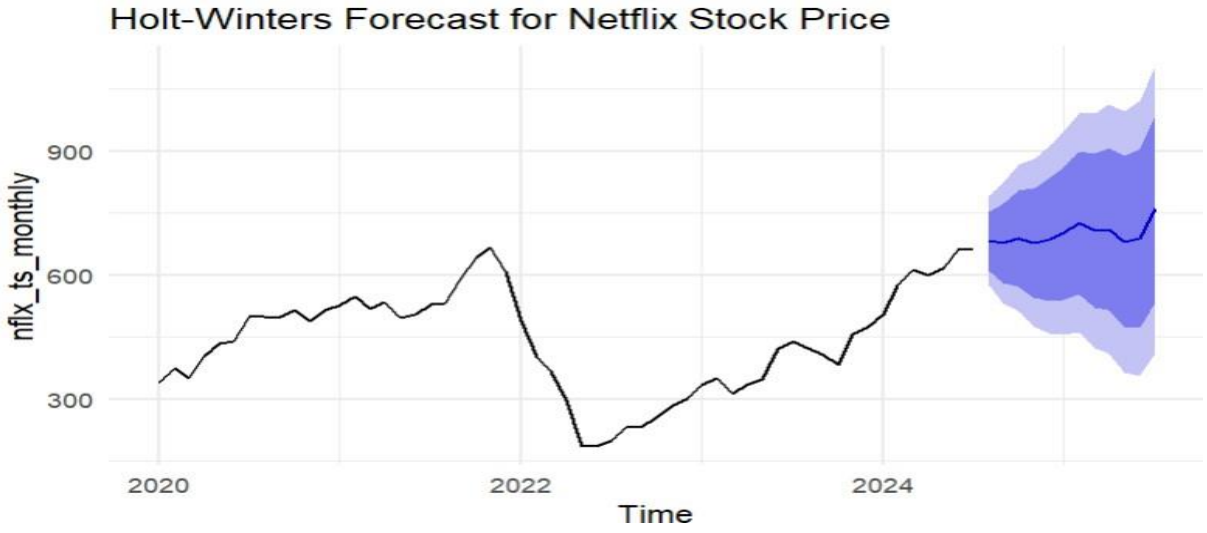
- R

```r
# Decompose the time series using additive model
decomp_additive <- decompose(nflx_ts, type = "additive")

# Decompose the time series using multiplicative model
decomp_multiplicative <- decompose(nflx_ts, type = "multiplicative")

# Plot the decomposed components for additive model
autoplot(decomp_additive) +
  ggtitle("Additive Decomposition of Netflix Stock Price") +
  theme_minimal()

# Plot the decomposed components for multiplicative model
autoplot(decomp_multiplicative) +
  ggtitle("Multiplicative Decomposition of Netflix Stock Price") +
  theme_minimal()
```

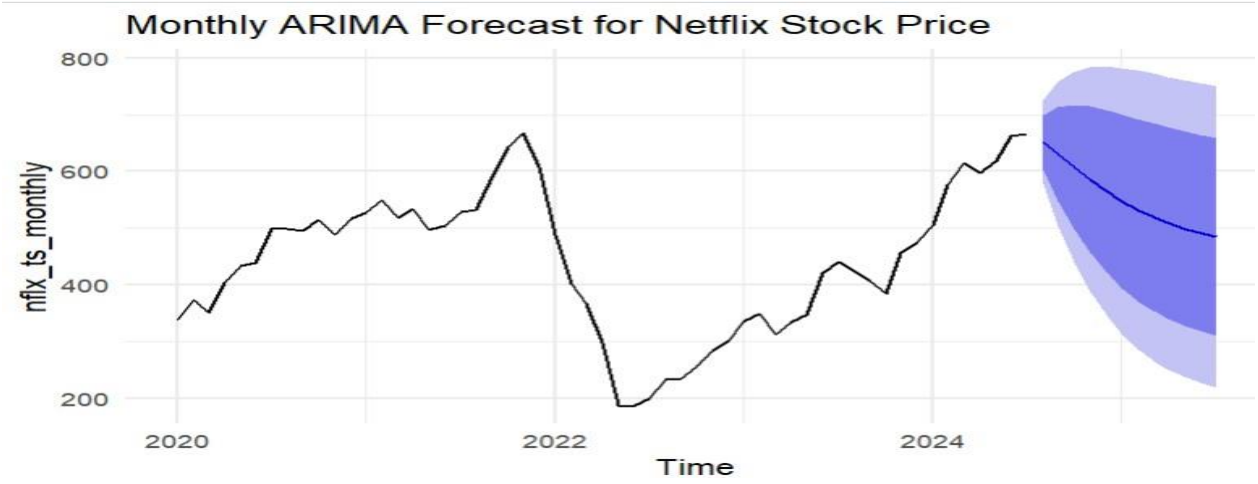





```r
# 1. Holt-Winters model and forecast for the next year
hw_model <- HoltWinters(nflx_ts_monthly)
hw_forecast <- forecast(hw_model, h = 12)
autoplot(hw_forecast) +
  ggtitle("Holt-Winters Forecast for Netflix Stock Price") +
  theme_minimal()

# 2. Fit ARIMA model to the daily data
arima_model_daily <- auto.arima(nflx_ts_daily)
summary(arima_model_daily)
```




```
Series: nflx_ts_daily
ARIMA(0,1,0)

sigma^2 = 140.4:  log likelihood = -4447.47
AIC=8896.94   AICc=8896.95   BIC=8901.98

Training set error measures:
                      ME     RMSE      MAE         MPE     MAPE        MASE
Training set -0.2647698 11.84228 7.861646 -0.1022131 1.935608 0.03416115
                    ACF1
Training set -0.02855902
```




```
> # Fit SARIMA model to the daily data
> sarima_model_daily <- auto.arima(nflx_ts_daily, seasonal = TRUE)
> summary(sarima_model_daily)
Series: nflx_ts_daily
ARIMA(0,1,0)

sigma^2 = 140.4:  log likelihood = -4447.47
AIC=8896.94   AICc=8896.95   BIC=8901.98

Training set error measures:
                     ME     RMSE      MAE         MPE     MAPE       MASE
Training set -0.2647698 11.84228 7.861646 -0.1022131 1.935608 0.03416115
                    ACF1
Training set -0.02855902
```

```
> # Compare ARIMA and SARIMA models
> arima_aic <- AIC(arima_model_daily)
> sarima_aic <- AIC(sarima_model_daily)
> print(paste("ARIMA AIC:", arima_aic))
[1] "ARIMA AIC: 8896.94154738081"
> print(paste("SARIMA AIC:", sarima_aic))
[1] "SARIMA AIC: 8896.94154738081"
```